UNIVERSIDADE FEDERAL DE VIÇOSA
DEPARTAMENTO DE INFORMÁTICA
INF221 – ENGENHARIA DE SOFTWARE I

PESQUISA PADRÕES DE PROJETO

OTÁVIO SILVA BITENCOURT
GUILHERME JOSE BITENCOURT LOPES
HENRIQUE RESENDE SILVA

VIÇOSA – MG
JULHO - 2024

Strategy

- O objetivo do padrão é parametrizar os algoritmos usados por uma classe. Ele prescreve como encapsular uma família de algoritmos e como torná-los intercambiáveis.
- Seu uso é recomendado quando uma classe é usuária de um certo algoritmo. Porém, como existem diversos algoritmos com esse propósito, não se quer antecipar uma decisão e implementar apenas um deles no corpo da classe.

Exemplo

- Estamos trabalhando em uma plataforma de E-Commerce que vende produtos eletrônicos e o código possui uma classe VENDA que é responsável pelas operações relacionadas à venda. A classe VENDA pode ter várias variantes, como uma venda de Natal ou uma Venda de Black Friday.
- A partir disso desejamos implementar a feature de aplicar descontos à venda. O desafio é que às formas de desconto não devem estar limitadas a um tipo específico de venda.

Implementação

```
interface DescontoStrategy {
    calculaDescontoTotal(itens: { item: string; preco: number}[]): number;
}

class CinquentaPorcentoDescontoStrategy implements DescontoStrategy {
    public calculaDescontoTotal(itens: { item: string; preco: number}[]): number {
        return (
            itens.reduce((soma, object) => {
                return soma + object.preco;
            }, 0) / 2
        );
    }
}

class PrimeiroItemDescontoStrategy implements DescontoStrategy {
    public calculaDescontoTotal(itens: { item: string; preco: number}[]): number {
        return (
            itens.reduce((soma, object) => {
                return soma + object.preco;
            }, 0) - itens[0].preco / 2
        );
    }
}
```

Figura 1 – Implementação da Interface e das diferentes estratégias

Implementação

```
class Venda {

    private descontoStrategy: DescontoStrategy;

    constructor(descontoStrategy: DescontoStrategy) {
        this.descontoStrategy = descontoStrategy;
    }

    public setStrategy(descontoStrategy: DescontoStrategy) {
        this.descontoStrategy = descontoStrategy;
    }

    public getDescontoTotal(itens: { item: string; preco: number}[]): void {
        const total = this.descontoStrategy.calculaDescontoTotal(itens);
        console.log(total);
    }
}
```

Figura 2 – Implementação da Classe Venda referenciando a interface criada anteriormente

Implementação

```
const carrinho = [
    { item: "Smartphone", preco: 1000},
    { item: "Notebook", preco: 2000}
]

const vendaBlackFriday = new Venda(new PrimeiroItemDescontoStrategy());
vendaBlackFriday.getDescontoTotal(carrinho);

vendaBlackFriday.setStrategy(new CinquentaPorcentoDescontoStrategy());
vendaBlackFriday.getDescontoTotal(carrinho);

const vendaNatal = new Venda(new CinquentaPorcentoDescontoStrategy());
vendaNatal.getDescontoTotal(carrinho);
```

```
Console ×

2500
1500
1500
```

Figura 3 – Utilização da Classe Venda e das diferentes estratégias

OBRIGADO!